# Programação Orientada a Objetos

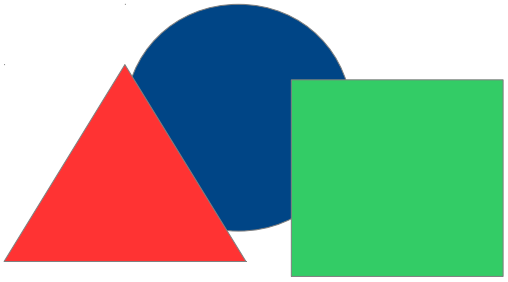
Nadia Félix/ Dirson S. Campos

UFG
UNIVERSIDADE
FEDERAL DE GOIÁS

# Programação Orientada a Objetos

OO

Orientação a Objetos



# Programação Orientada a Objetos

A **orientação a objetos** é um **paradigma** de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

A **orientação a objetos** visa um pensamento o mais próximo possível da vida real.

# Programação Orientada a Objetos

**O que é um Paradigma?**

- Um modelo, padrão, estilo, protótipo
- É a representação de um padrão a ser seguido
- Formas de abstração do problema
- Estilo de programação

*Define a forma como um desenvolvedor lida com um problema – tanto na análise como na programação*

# Programação Orientada a Objetos

**Paradigma Imperativo**

- *Primeiro faça isso, depois faça aquilo*

  uma sequência de comandos que o computador executará, passo a passo, modificando dados e variáveis a fim de chegar ao resultado esperado (Pascal, Cobol, C, etc).

- Derivados : Estruturado ou Procedural

# Programação Orientada a Objetos

**Paradigma Funcional**

- Subdividir o problema em funções matemáticas e resolver cada uma separadamente, pois os resultados encontrados serão utilizados posteriormente.

  Ex: Lisp, R, Erlang

# Programação Orientada a Objetos

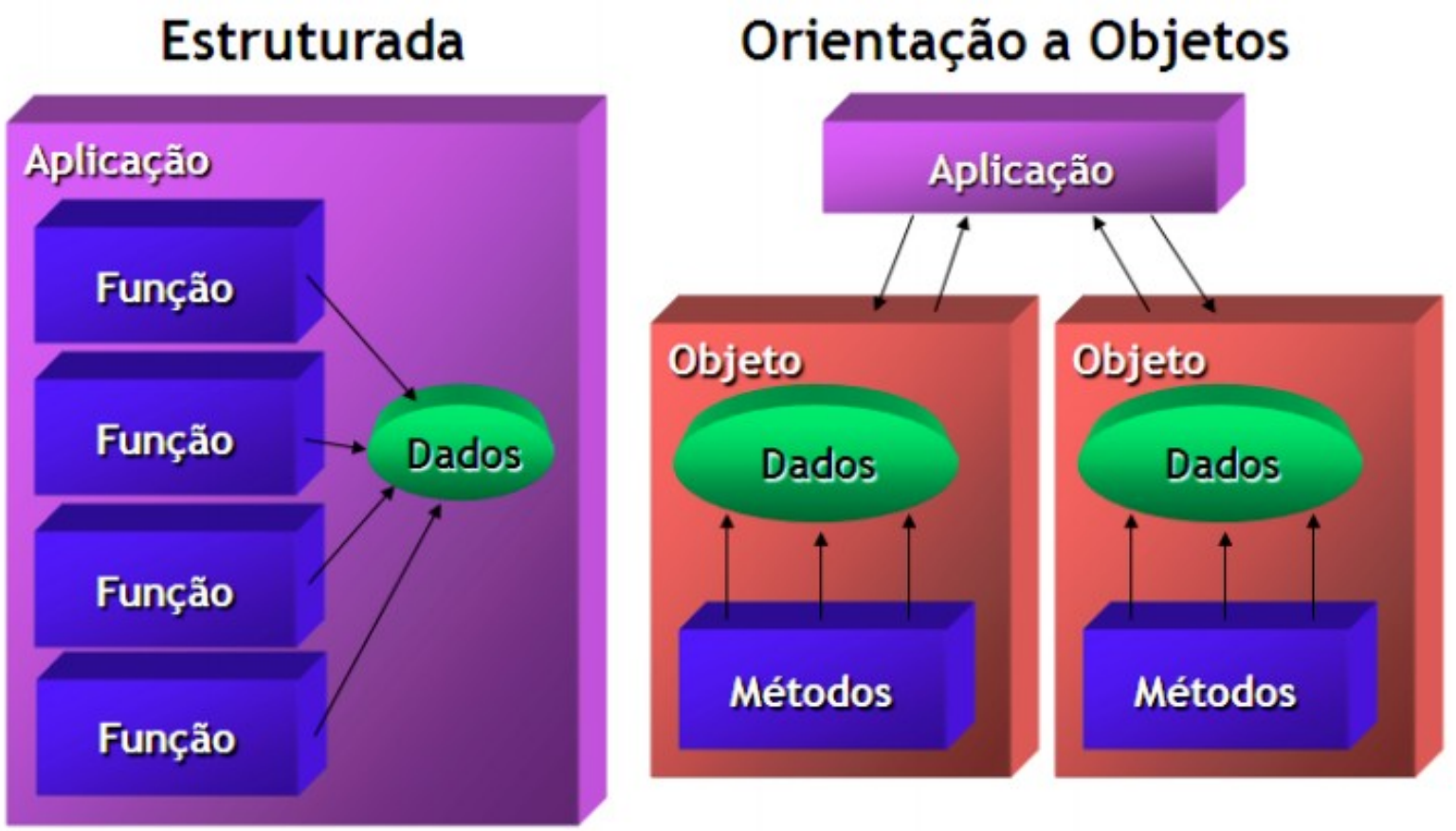
**Paradigma Orientado a Objetos**

- Um conjunto de **classes** (estruturas definidas) que gera **objetos** (instâncias) e, estes recebem ordens para executar tarefas através de troca de mensagens.
- O Software como um todo se forma com a interação entre os objetos através das trocas de mensagens.

Vantagens: manutenção, flexibilidade e reutilização

# Programação Orientada a Objetos

**Diferença entre os principais paradigmas**



# Programação Orientada a Objetos

**Diferença entre os principais Paradigmas**

**Sequência de Passos x Objetos**

- Programação como sequência de passos
  - Paradigma estruturado (tradicional) onde um problema é resolvido a partir de um início e fim bem definidos e eventualmente dividido em sub-rotinas;
- Programação utilizando objetos
  - O paradigma da orientação a objetos considera que os dados a serem processados e os mecanismos de processamento dos dados (operações) devem ser considerados em conjunto;

# Programação Orientada a Objetos

**Um pouco de história sobre POO**

**(Programação Orientada a Objetos)**

- (1967) SIMULA - 1ª Linguagem Orientada a Objetos;
- Década de 70 surge a linguagem SmallTalk (considerada puramente orientada a objetos);
- Década de 80 ... Rápida evolução ... Surgimento de Ada e C++
- Década de 90 ... Java

# Programação Orientada a Objetos

## ..mais história

- O uso da Tecnologia de Objetos como metodologia básica para desenvolvimento de sistemas (abrangendo todo o ciclo ... desde análise até o código) é uma prática que passou a ser difundida na década de 80 com a publicação dos trabalhos do pesquisador Grady Booch.
- A tecnologia de objetos veio para ficar!
- Seus conceitos e técnicas imprimem maior qualidade, produtividade e profissionalismo na construção de software

# Programação Orientada a Objetos

## Orientação a objetos é necessária?

- Nem sempre ...
- Há situações onde o modelo de uma tarefa a ser executada é tão simples que a criação de uma classe para representá-lo torna o problema mais complicado e confuso ...
- Exemplo: calcular as raízes de uma equação de segundo grau

$$x = \frac{-b \mp \sqrt{b^2 - 4\mathrm{ac}}}{2\mathrm{a}}$$

Fórmula de Bhaskara

# Programação Orientada a Objetos

```c
#include <stdio.h>
#include <math.h>

double calculaX1(int a, int b, int c) {
   double delta = (b*b)-4*(a)*(c);
   return -b + sqrt(delta) / 2*a;
}

double calculaX2(int a, int b, int c) {
   double delta = (b*b)-4*(a)*(c);
   return -b - sqrt(delta) / 2*a;
}

int main(int argc, char** argv)  {
   int a, b, c;
   printf("Digite a, b e c ... \n");
   scanf("%d",&a);
   scanf("%d",&b);
   scanf("%d",&c);
   printf("X1 = %f", calculaX1(a, b, c));
   printf("X2 = %f", calculaX2(a, b, c));
   return 0;
}
```

**Código estruturado em C**

```java
public class Bhaskara {
   private int a, b, c;

   Bhaskara(int a, int b, int c) {
      this.a = a;
      this.b = b;
      this.c = c;
    }

   public double calculaX1() {
      double delta = (b*b)-4*(a)*(c);
      return -b + Math.sqrt(delta) / 2*a;
   }

   public double calculaX2() {
     double delta = (b*b)-4*(a)*(c);
     return -b - Math.sqrt(delta) / 2*a;
   }

   public static void main (String args[]) {
      Bhaskara bascara = new Bhaskara(3,-7,2);
      System.out.println(bascara.calculaX1());
     System.out.println(bascara.calculaX2());
  }
}
```

**Código orientado a objetos em Java**

# Programação Orientada a Objetos

```java
import java.util.Scanner;

public class CalculoIMC {

public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Entre o nome da pessoa:");
  String nome = sc.nextLine();
  System.out.println("Entre a idade:");
  int idade = Integer.parseInt(sc.nextLine());
  System.out.println("Entre a altura:");
  double altura = Double.parseDouble(sc.nextLine());
  System.out.println("Entre a peso:");
  double peso = Double.parseDouble(sc.nextLine());

  double imc = peso / (altura*altura);

  System.out.println("Nome: " + nome + " - Idade: "          +
idade + " - Altura: " + altura + " - Peso: "        + peso + " - IMC:
" +    imc);
  }
}
```

**Código estruturado em Java**

```java
public class Pessoa {
   String nome;
   int idade;
   double peso;
   double altura;

   Pessoa(String nome, int idade, double peso, double             altura)
{
     this.nome = nome;
     this.idade = idade;
     this.peso = peso;
     this.altura = altura;
       }

   public double IMC() {
     return peso/(altura * altura);
       }
}

public class OutraClasseQualquer {
   public static void main(String[] args) {
     Pessoa pes = new Pessoa("Maria",20,60,1.53);

     System.out.println("Nome: " + pes.nome + " - Idade: "       +
pes.idade +  " - Altura: " + pes.altura + " - Peso:     " + pes.peso + " -
IMC: " + pes.IMC());
     }
}
```

**Código orientado a objetos em Java**

# Programação Orientada a Objetos

## Orientação a objetos é necessária?

- Em muitas situações ...
- Imagine uma aplicação “mundo real” repleta de janelas que apresentam as mesmas funcionalidades e se compõem de uma infinidade de outros controles gráficos que se repetem ao longo da aplicação (botões, caixas de texto, janelas de diálogo...);
- Imagine simular o movimento de espermatozoides a procura de um óvulo utilizando a programação estruturada normal ... De forma que cada célula (dentre as milhões) tenha atributos diferentes (velocidade, tamanho, agilidade...)

# Programação Orientada a Objetos

**Vantagens da Orientação a Objetos:**

- Ajuda na organização e resolve problemas da programação procedural/estruturada
- Minimiza o código - escreve-se menos
- Em alguns casos facilita o entendimento
- Concentra as responsabilidades nos pontos certos
- Diminui a responsabilidade dos programadores

# Programação Orientada a Objetos

**Aplicação**

- **Resolução de Problemas**

Termos semelhantes que serão muito utilizados :

- Problema
- Domínio
- Contexto
- Cenário
- Situação

# Programação Orientada a Objetos

**Exemplo:**

Cenário: inscrição de um aluno em uma disciplina

- Como seria no paradigma estrutural?

- Como seria no paradigma orientado a objetos?

# Programação Orientada a Objetos

**Exemplo:**

Cenário: Saque de uma conta corrente de um cliente

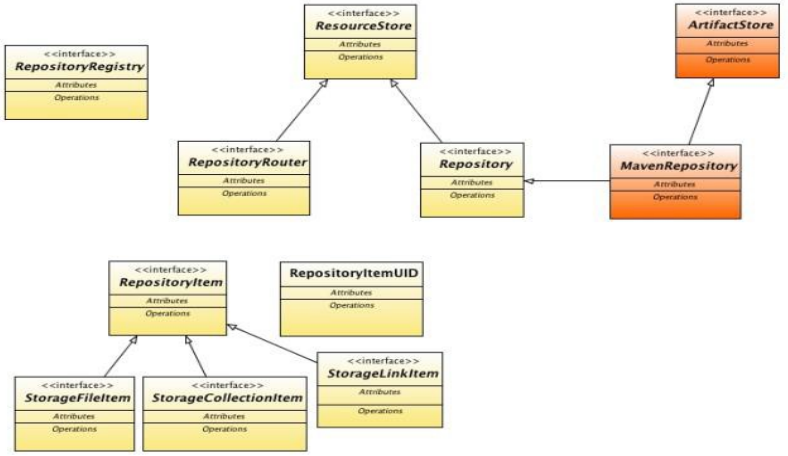
- Como seria no paradigma estrutural?

- Como seria no paradigma orientado a objetos?

# Programação Orientada a Objetos

## POO

Conceitos Fundamentais



# Programação Orientada a Objetos

Uma linguagem orientada a objetos deve oferecer:

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

De acordo com...

Grady Booch: Coming of Age in an Object-Oriented World. IEEE Software 11(6): 33-41 (1994).

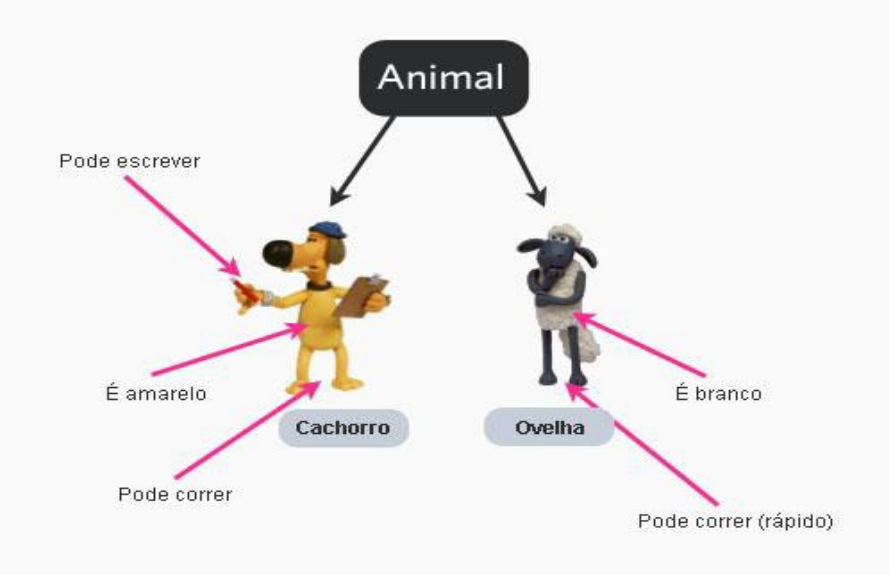
# Programação Orientada a Objetos

## ABSTRAÇÃO

CLASSES X OBJETOS

- Habilidade de se concentrar nos aspectos essenciais de um **contexto** qualquer, ignorando características menos importantes ou acidentais;

- Imaginar quais são os **objetos** e o que eles vão realizar no sistema



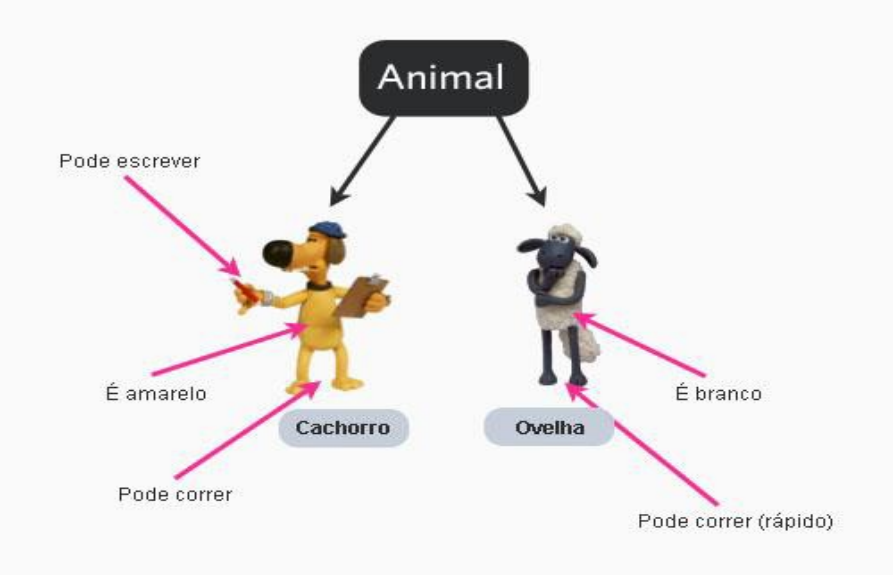
# Programação Orientada a Objetos

## ABSTRAÇÃO

CLASSES X OBJETOS

- Na POO, uma **classe** é uma abstração de entidades existentes no **domínio** do sistema de software;

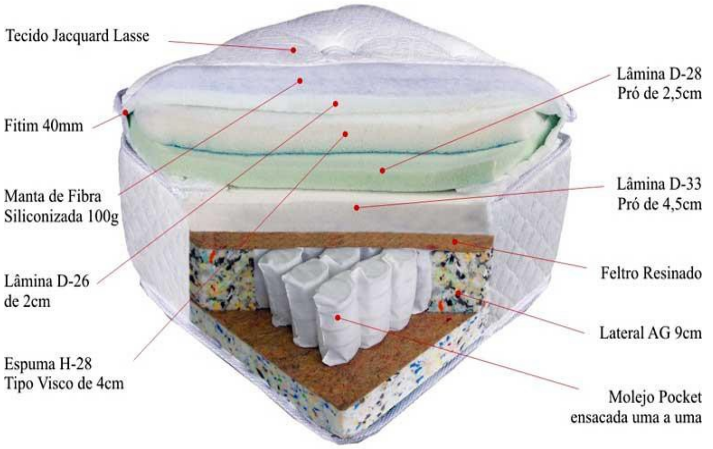
- Os **objetos** são instâncias das classes e cada um tem sua identidade, propriedades e comportamentos.



# Programação Orientada a Objetos

## ENCAPSULAMENTO

Ocultação de Informações



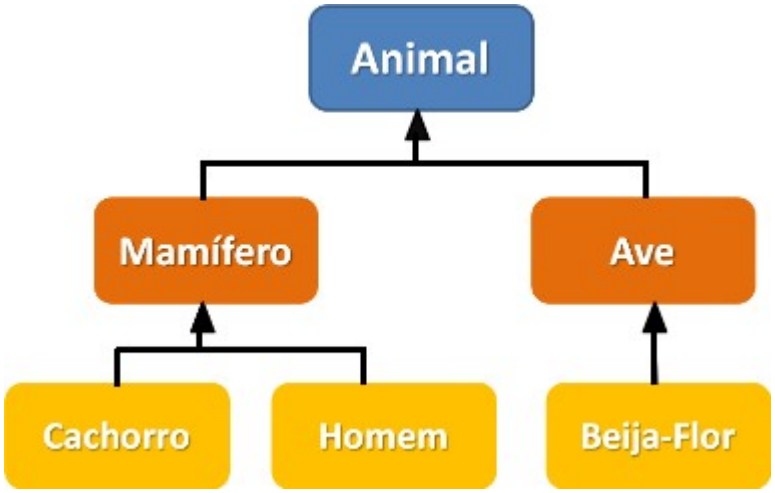

- Adicionam **segurança** à aplicação impedindo o acesso direto ao estado de um objeto;
- Consiste na separação dos aspectos internos e externos de um objeto;
- Permite ignorar os detalhes de implementação (de como as coisas funcionam internamente) permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração;

# Programação Orientada a Objetos

## HERANÇA

Reutilização de Código

- É o mecanismo de reaproveitamento e **reutilização de código**;
- Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos;

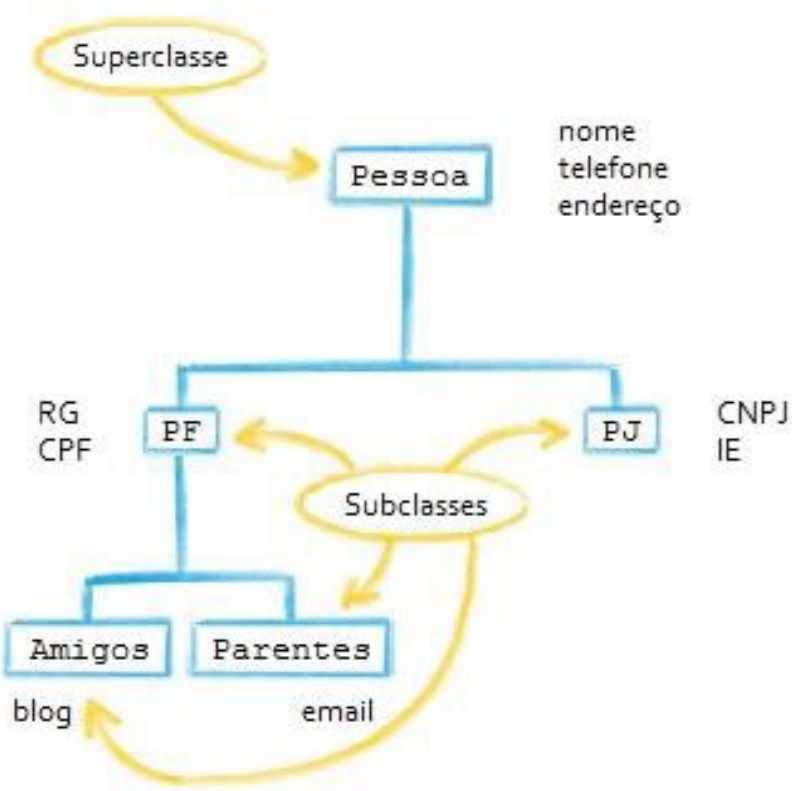


# Programação Orientada a Objetos

## HERANÇA

Reutilização de Código

- Podem ser diretas e indiretas e as estruturas são formadas para que algumas **Superclasses** agrupem características e comportamentos mais genéricos que possam ser reaproveitados por **Subclasses** mais específicas

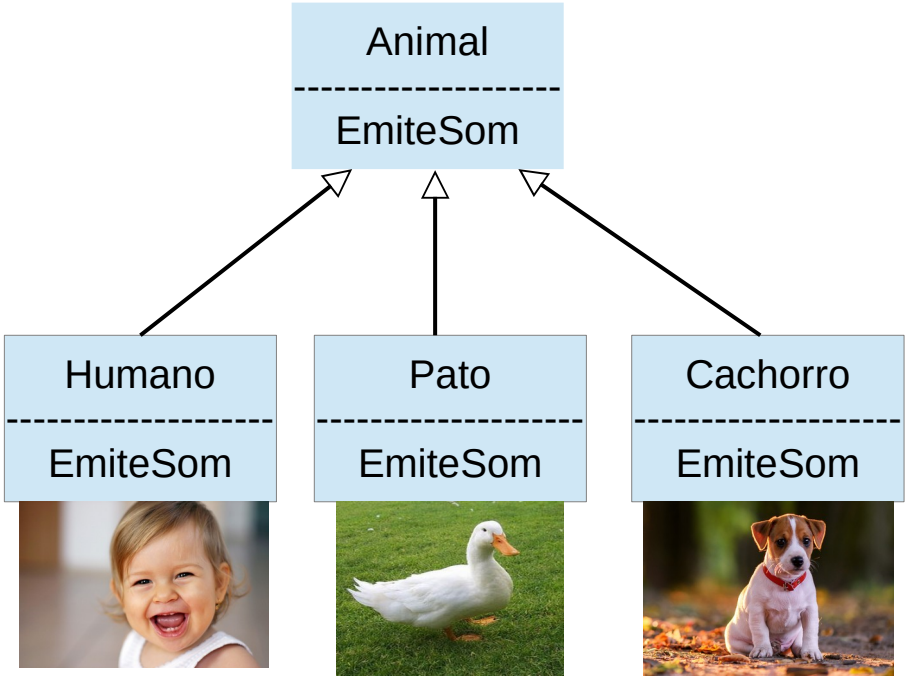


# Programação Orientada a Objetos


## POLIMORFISMO

Várias Formas

- Permite a um mesmo objeto se manifestar de diferentes formas;
- Implementações diferentes para uma mesma operação.



# Programação Orientada a Objetos

## Exemplos e  Exercícios

Entendo as diferenças entre programação estruturada e orientada a objetos